5.4 The Lunar Lander

Author: Michele Moro

5.4.1 Teacher's guide

- Title: the Lunar Lander
- Introduction: Simulation of the landing of a lunar module, assumed to compensate partly the lunar gravitation with a reaction engine.
- Goals:
    - To improve the knowledge of some basic physical items like space, speed, acceleration, time
    - To study the theory of basic motions, like the uniformly accelerated (decelerated) motion;
    - To solve first grade equations and systems;
    - To afford the difficulties of the landing problem with a simple and robust sensor-based robotic solution.
- Age group: 16-19 years old.
- Rationale of the teaching approach: physics is better taught and learned when theory is presented together with some experimental activities. There are several well known approaches to present basic items like space, time etc., but some suffer of the lack of an easy reproducibility in normal conditions or have no clear relation to normal life experiences. The lunar landing problem can be easily presented and understood in its essence even though its solution is related to a theory with evident difficulties. The presented robotic simulation can help the students in understanding such a solution and can make them more comfortable with other more general similar theories.

5.4.2 The problem

From the Kepler and Newton laws we know that the motion of an object subjected to the gravitational force of a planet or a satellite is in general a conic, but as a special case of a body that has a null initial speed or whose initial velocity vector points towards the center of the mass of the planet, the motion tends to the degenerate case of a motion following a line, and the body sooner or later will fall on the surface of the planet. If a lunar module should 'land' on the surface of our satellite coming from the Earth, to prevent the destructive impact, it is necessary to impress, through a jet engine, a force in the direction opposite to the attraction, that is, tending to push it out of the Moon, that not only compensates for the

acceleration, but makes the objects reduce their speed near the surface to very small values, essentially under a negligible threshold (fig. 5.4.1).

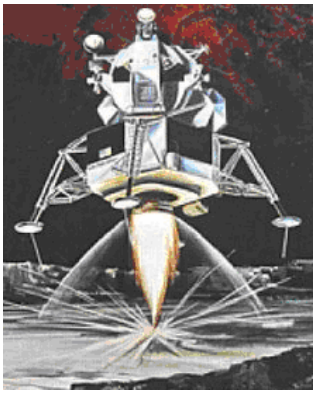

Fig. 5.4.1 – *A lunar module*

To give an idea of the speed you can reach close to the surface of a planet or a satellite, we assume that a mass m is 'dropped' with speed zero 1 km away from the lunar surface, and that there are no other forces different from the lunar gravitation (no friction, no influence of other bodies). The acceleration, which in this case is continuously parallel to the motion and oriented towards the moon, has the effect of increasing the speed, and its modulus is equal to:

$$a_G = F_G / m = G\,M / r^2 = \mu / r^2 \qquad (5.4.1)$$

where G is the universal gravitational constant, M the mass of the Moon, and r the distance from the center of the planet/satellite, which serves as a center of mass. In the case of the Moon we have:

$$\mu_{MOON} = G\,M_{MOON} = 6.670\ 10^{-11} \cdot 7.34\ 10^{22} = 4.895\ 10^{12}\ Nm^2kg^{-1} \qquad (5.4.2)$$

With the data of the problem, the initial acceleration is:

$$a_{G0} = \mu_{MOON} / ((R_{MOON} + 10^3)^2) \qquad (5.4.3)$$

Using the average radius of the Moon, corresponding to $1.74\ 10^6$ m, as the distance of the surface from the center of mass, we see that in the considered range of distances, the acceleration increases slightly approaching to the lunar surface. In fact, at the beginning and end of motion it holds:

$$a_{G0} = 4.895\ 10^{12} / ((1.74\ 10^6 + 10^3)^2) = 1.6149$$
$$a_{Gf} = 4.895\ 10^{12} / 1.74\ 10^{12} = 1.6167 \qquad (5.4.4)$$

Because for an elliptical orbit it holds:

$$v = \mathrm{sqrt}(\mu\ ((2/r)-(1/a))) \qquad (5.4.5)$$

giving the speed value when the distance varies in the elliptical (closed) motion, we obtain that in order to justify the initial null speed it must hold:

$$(2/r_0)=(1/a) \qquad a = r_0/2 \qquad (5.4.6)$$

that is, it is actually a degenerate motion in which we can consider the eccentricity of the ellipse tending to 1, with the body leaving at null speed from the secondary focus and moving to reach the attractive focus. With our data, (5.4.5) and (5.4.6), we give the speed at the impact on the surface:

$$v = \mathrm{sqrt}(\mu_{MOON}\ ((2/R_{MOON})-(2/(R_{MOON}+10^3)))) = 56.84\ ms^{-1} = 204.65\ km\ h^{-1} \qquad (5.4.7)$$

a considerable speed, certainly destructive.

5.4.3 The theory

Let us assume that in our simulation we can easily control the speed of the robot: this is true because the settable power of the motors of the NXT is actually an angular speed control, at least until such control can compensate for a possible resistant torque. So, wanting to provide an analytical solution of the problem, a first approach is to set 'a priori' to a specific speed time profile (v = v (t)) that has the desired characteristics, in particular a desired initial speed and a final speed tending to zero.

The profiles of this kind are numerous: we will conduct our analysis in a simple case, a uniformly decelerated motion (fig. 5.4.2).

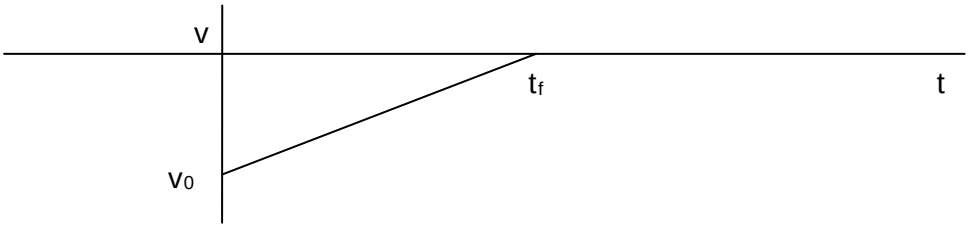

Fig. 5.4.2 – *Uniformly decelerated motion*

In this case it holds:

$$v(t) = v_0 - (v_0 / t_f)\cdot t \qquad (5.4.8)$$

v_0 is negative and acceleration is constant (positive) and equal to $-v_0/t_f$, and $v(t_f)=0$. But how much is t_f, at the moment when the motion is completed and the body is on the surface? This parameter depends on the spatial motion and it is the moment where $r(t_f) = R$ (radius of the planet/satellite). This motion is given by:

$$r(t) = r_0 + {}_0\int^t v(t)\cdot dt \qquad (5.4.9)$$

A primitive function of (5.4.8) is:

$$\rho(t) = v_0\cdot t - (v_0 / (2\cdot t_f))\cdot t^2 \qquad (5.4.10)$$

so from (5.4.9) we obtain:

$$r(t_f) = R = r_0 + [v_0\cdot t - (v_0 / (2\cdot t_f))\cdot t^2]_0^{tf} = r_0 + v_0\cdot t_f - (v_0 / 2)\cdot t_f \qquad (5.4.11)$$

$$t_f = 2\cdot(R-r_0) / v_0 = -2\cdot(r_0-R) / v_0 \qquad (5.4.12)$$

(remember that v_0 is negative). On the numerator there is the initial distance from the surface. With this value of the t_f the (5.4.8) and (5.4.9) become:

$$v(t) = v_0 + (v_0^2 / (2(r_0-R)))\cdot t \qquad (5.4.13)$$

$$r(t) = r_0 + v_0\cdot t + (v_0^2 / (4\cdot(r_0-R)))\cdot t^2 \qquad (5.4.14)$$

If, for example, we complete the approaching phase in 1 minute, starting at 1 km far from the surface, it should be:

$$v_0 = -2\cdot(r_0-R) / t_f = -2\cdot 10^3 / 60 = -100/3\ ms^{-1} = -120\ kmh^{-1} \qquad (5.4.15)$$

from which it derives as a (positive) deceleration:

$$a = v_0 / t_f = 100 / (3\cdot 60) = 0.555\ ms^{-2} \qquad (5.4.16)$$

a very limited value, when compared with the terrestrial acceleration of gravity (9.78).

Assuming that the students have the knowledge to derive the mathematical description of the motion based on the chosen speed profile, the approach presented here can in principle be adopted by the simulation using the NXT robot, but the accuracy in the implementation of the speed profile is essentially tied on maintaining the accuracy of the time going by, which maybe somewhat problematic for the NXT. Moreover, no sensorial capability of the robot is exploited that could make it possible to realize a 'robust' control that adapts itself to the inevitable inaccuracies.

The NXT kit includes a sensor of distance that gives us the opportunity to adopt an alternative approach, where the user defines a speed/space profile ($v = v(s)$) instead of a speed/time one. Now we conduct the analysis again on a seemingly simple case, a linear profile (fig. 5.4.3)

The profile is reasonable because in the initial position (r_0) the initial velocity is v_0 and when the body reaches the surface, the speed is zero. Now how can we derive the temporal functions of space, velocity and acceleration?

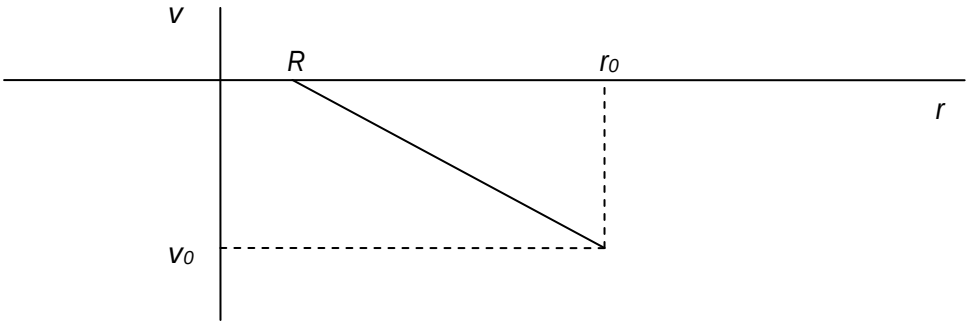

Fig. 5.4.3 – *Linear speed/space profile*

For the given profile it holds:

$$v(r) = (v_0 / (r_0\text{-}R))\cdot(r\text{-}R) = dr/dt \qquad (5.4.17)$$

(5.4.17) is a first order differential equation, a mathematical relation usually affordable at the university level, and usually too harder for a student of secondary education (though some secondary schools present it during the last year). Fortunately, as we shall see in the next section, making the robot to follow a speed profile does not require to solve analytically such a relation (or a similar one), but for the sake of completeness we recall that such kind of equation as a general solution the family of exponential functions. The solution, applying the constraints of our case, is:

$$r(t) = R + (r_0\text{-}R)\, e^{(v0 / (r0\text{-}R))\cdot t\cdot} \qquad (5.4.18)$$

The function in (5.4.18) tends to *R* when $t\to\infty$: therefore, the motion is absolutely a not uniformly decelerated motion, as a superficial observer could believe in, and in theory the surface is really reached only asymptotically. This is not a problem for the NXT because the motor has a minimum speed to be taken into account, as well as the precision of the distance sensor. Calculating the derivative to obtain the the temporal profile of velocity, we obtain:

$$v(t) = (r_0\text{-}R)\,(v_0 / (r_0\text{-}R))\, e^{(v0 / (r0\text{-}R))\cdot t\cdot} \qquad (5.4.19)$$

Thus, even the speed decreases exponentially.

Also, in theory, we try now to calculate what speed/space profile would produce a uniformly decelerated motion, through a reasoning inverse of that made above. From (5.4.8), assuming $a_0 = -v_0/t_f = v_0^2 / (2\,(R\text{-}r_0))$, we obtain:

$$t(v) = (v\text{-}v_0) / a_0 \qquad (5.4.20)$$

Substituting now in (5.4.14) it holds:

$$r(v) = r(t(v)) = r_0 + v_0\cdot(v\text{-}v_0) / a_0 + (a_0 / 2)\,((v\text{-}v_0)/a_0)^2 \qquad (5.4.21)$$

$$r(v) = r_0 + (v_0 \cdot / a_0) \cdot v - v_0^2 \cdot / a_0 + (1/(2 \cdot a_0))(v^2 - 2v_0 v + v_0^2) \qquad (5.4.22)$$

Moving *r* to the second member and multiplying by $2a_0$ we obtain the equation of second degree:

$$v^2 - v_0^2 + 2a_0 r_0 - 2a_0 r = 0 \qquad (5.4.23)$$

Once solved, getting the negative solution, as the speed must be, the (5.5.23) gives the following profile (always taking into account that v_0 is negative):

$$v(r) = -\text{sqrt}(2a_0 r + v_0^2 - 2a_0 r_0) = -\text{sqrt}((v_0^2 / (r_0 - R)) \cdot r + v_0^2 - v_0^2 r_0 / (r_0 - R)) = -\text{sqrt}((v_0^2 / (r_0 - R)) \cdot r - v_0^2 R / (r_0 - R)) = (v_0 / \text{sqrt}(r_0 - R)) \cdot \text{sqrt}(r - R) \qquad (5.4.24)$$

To check the result, it is known that $v(r_0) = v_0$ and $v(R) = 0$. Thus, concluding, the speed/space profile that produces a uniformly decelerated motion is a scaled and shifted square root (fig. 5.4.4).

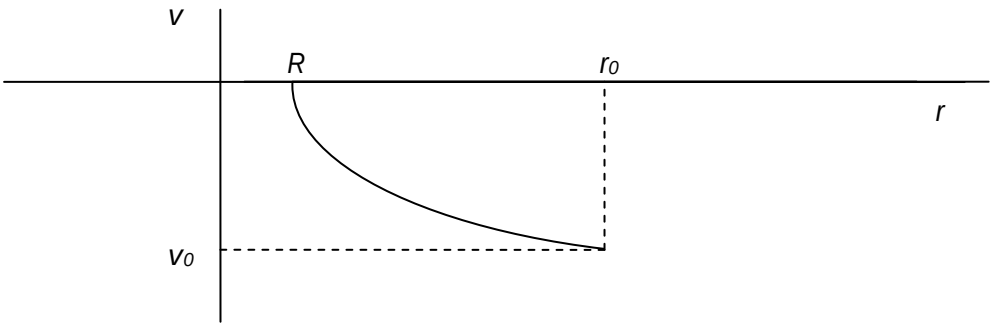

Fig. 5.4.4 – *Speed/space profile for a uniformly decelerated motion*

5.4.4 Our simulation with NXT

Taking the second of the two approaches presented, always within the precision that qualifies NXT, the simulation is pretty simple: we just mount the sonar sensor on the robot and make the applied power depend on the measured distance according to the chosen speed/space profile. To represent the descent of a lunar module, we decided to mount a single motor on the robot that, appropriately de-multiplied, unrolls from a top spool a tape to which the robot is hung to a fixed point (fig. 5.4.5). The speed is controlled as usual by the 'power' parameter of the motor. We have empirically determined the minimum power under which the robot is still: that threshold depends also on the action of the resistant torque caused by the weight of the robot. This parameter must follow the speed profile determined

by the measure of the distance from the ground provided by the sonar, which is oriented downwards.

Fig. 5.4.5 – *The lunar module*

Another inaccuracy, especially if the spool has a small radius, is that the radius of unrolling is not constant and, thus, the speed of the robot is not exactly proportional to the angular speed of the motor. Again, this imprecision is self-compensated thanks to the measure of the distance from the ground repeated at the maximum possible frequency, and, in particular, it is certain that, when the robot is very close to the ground, the speed is very low and the impact smooth.

The code is divided into an initial phase and a loop. The initial phase settles the initial speed and distance (variables v_0 and x_0) and checks if the initial position is too small (less than 10), in which case it stops the program. Space is measured in cm and speed in cm/s (fig. 5.4.6).

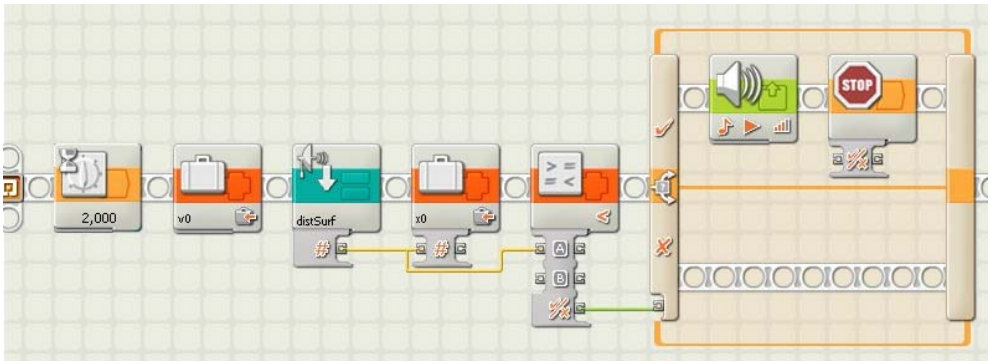

Fig. 5.4.6 – *The Initial phase*

VarDecl(Name=v0, Type=NUM) -- the initial speed

VarDecl(Name=x0, Type=NUM) -- the initial position

VarDecl(Name=v, Type=NUM) -- speed

```
VarDecl(Name=x, Type=NUM) -- position

Wait(Ctrl=TIME, Until=2)

Var(Name=v0.NUM, Act=WR, Val=50)

Ds1:distSurf()

Var(Name=x0.NUM, Act=WR, Val=Ds1.Result)

Cm1:CmpOp(Type=LT, A=Ds1.Result, B=10)

Sw1: Switch(Ctrl=VAL, Type=LOGIC, Dis=ON, CondUp=TRUE,
        Val=Cm1.Res)

[Sw1.IF

        Sound(Act=TONE, Ctrl=PLAY, Vol=75, Rep=OFF, Note=C2,
         Dur=1, Wait=ON)

        Stop()

Sw1.IF]

[Sw1.ELSE

Sw1.ELSE]
```

The *distSurf* block returns the corrected distance from the landing surface, which takes into account the offset of the sonar in respect of the 'feet' of the landing module (fig. 5.4.7).

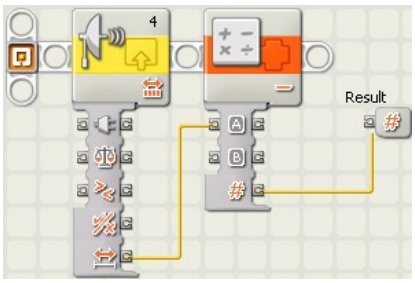

Fig. 5.4.7 – *The distSurf MyBlock*

```
MyBlock(Name=distSurf, InParams=(), OutParams=(Result.NUM)) {
        Ss1:SonarSens(Port=4, Cmp=??, Show=CM)
        Su1:MathOp(Type=SUB, A=Ss1.Dist, B=<offset>)
        SetOut(Name=Result.NUM, Val=Su1.Res)
distSurf}
```

The second part is the main loop, which is responsible for calculating the current speed to be applied to the motor. After having taken the measure of the distance, the code within the loop calculates the corresponding speed in accordance with the speed/space profile, linear in this case. Because the distance is corrected so that it is equal to 0 when the landing module reaches the lunar surface, the profile is simply:

$$v(x) = (v_0 / x_0) \cdot x \qquad (5.4.25)$$

Current distance and speed are assigned respectively to variables x and v. We obtain the code of fig. 5.4.8.

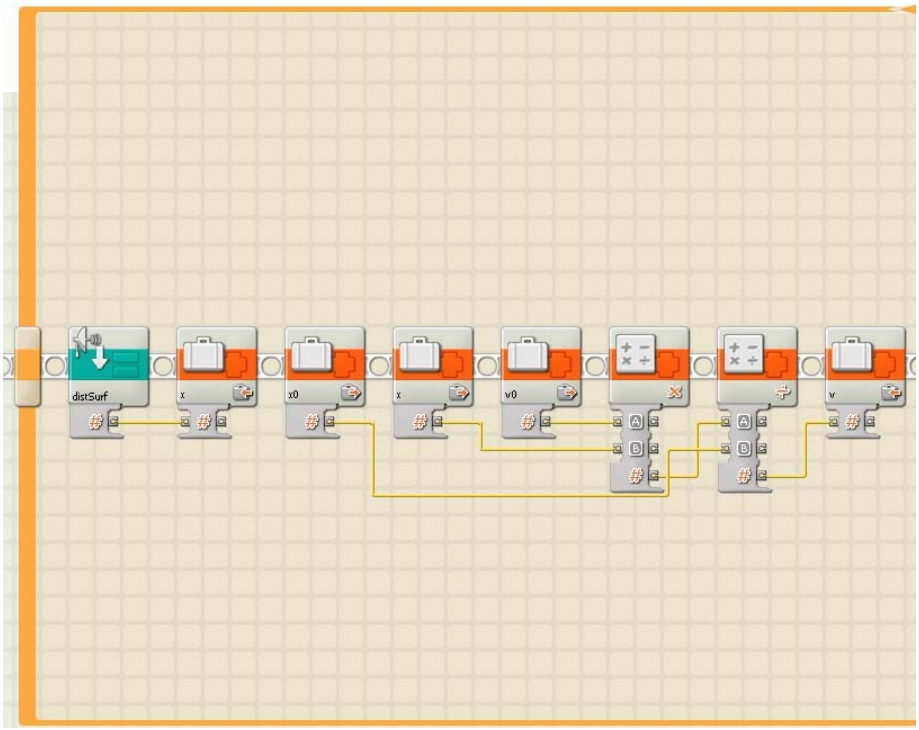

Fig. 5.4.8 – *The main loop (part I)*

Loop1: Loop(Ctrl=FOREVER, Dis=OFF) [

Ds2: distSurf()

Var(Name=x.NUM, Act=WR, Val=Ds2.Result)

Vd1:Var(Name=x0.NUM, Act=RD)

Vd2:Var(Name=x.NUM, Act=RD)

Vd3:Var(Name=v0.NUM, Act=RD)

Mu1:MathOp(Type=MUL, A=Vd3.Val, B=Vd2.Val)

Di1:MathOp(Type=DIV, A=Mu1.Res, B=Vd1.Val)

Var(Name=v.NUM, Act=WR, Val=Di1.Res)

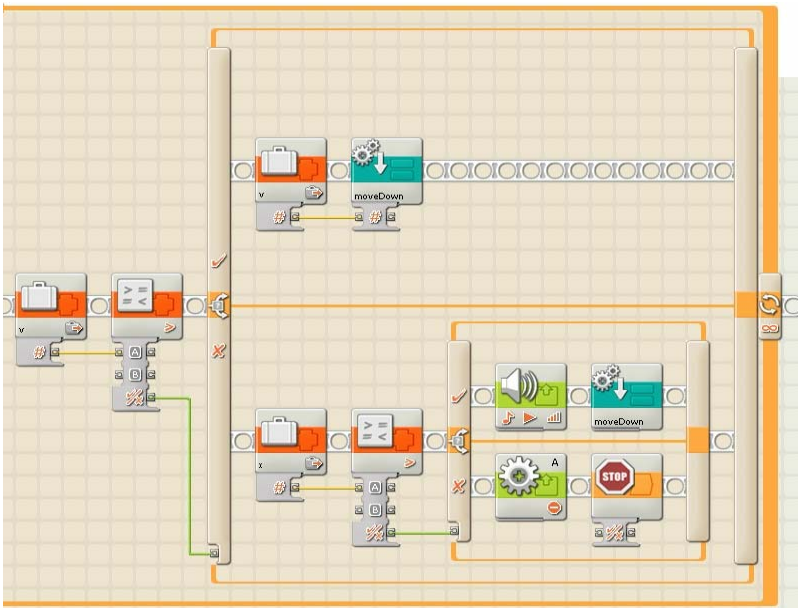

Fig. 5.4.9 – *The main loop (part II)*

Now, we distinguish two cases, one when the vehicle is still far from the surface and makes a normal motion step with the calculated speed; the second is necessary, when the vehicle is very close to surface. Due to the integer calculation limit, the speed could result in 0 even though the vehicle is still going down very slowly (remember the exponential motion). Thus, for practical reasons, we manage this situation by applying a constant speed of 1 until the distance becomes 0: only at that moment we stop the motion (fig. 5.4.9).

```
        Vd4:Var(Name=v.NUM, Act=RD)

        Cm2:CmpOp(Type=GT, A=Vd4.Val, B=1)

        Sw2: Switch(Ctrl=VAL, Type=LOGIC, Dis=ON,
          CondUp=TRUE, Val=Cm2.Res)

        [Sw2.IF

                Vd5:Var(Name=v.NUM, Act=RD)

                moveDown(v=Vd5.Val)

        Sw2.IF]

        [Sw2.ELSE

                Vd6:Var(Name=x.NUM, Act=RD)

                Cm3:CmpOp(Type=GT, A=Vd6.Val, B=0)

                Sw3: Switch(Ctrl=VAL, Type=LOGIC, Dis=ON,
                  CondUp=TRUE, Val=Cm3.Res)

                [Sw3.IF

                        Sound(Act=TONE, Ctrl=PLAY, Vol=75,
                          Rep=OFF, Note=B4, Dur=0,5, Wait=OFF)

                        moveDown(v=1)

                Sw3.IF]

                [Sw3.ELSE

                        Motor(Port=A, Dir=STOP, Next=BRK)

                        Stop()

                Sw3.ELSE]

        Sw2.ELSE]

Loop1]
```

The *moveDown* block scales the imposed speed so that the range of speeds (1÷50) corresponds to the range of powers (20÷100). The calculation of the parameters α and β of the linear function $power = \alpha \cdot speed + \beta$ is a good exercise, being the resolution of a system of two first grade equations:

$$\begin{cases} 20 = \alpha \cdot 1 + \beta \\ 100 = \alpha \cdot 50 + \beta \end{cases} \qquad (5.4.26)$$

Subtracting the first from the second you obtain:

$$\begin{cases} 80 = \alpha \cdot 49 \quad \Rightarrow \quad \alpha = 80/49 \\ \beta = 20 - \alpha = (980 - 80)/49 = 900/49 \end{cases} \qquad (5.4.27)$$

$$power = (speed \cdot 80 + 900) / 49 \qquad (5.4.28)$$

The imposed speed is the *v* formal parameter of the *moveDown* block (fig. 5.4.10).

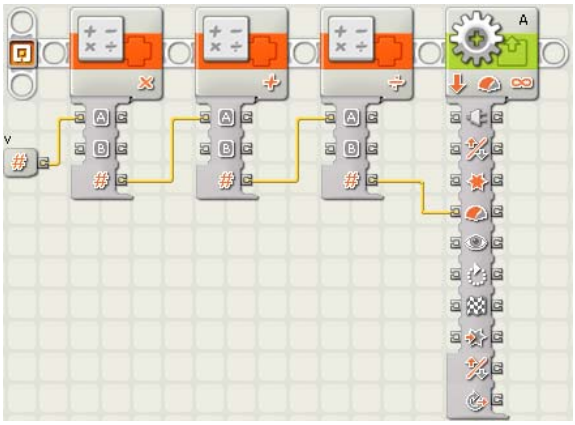

Fig. 5.4.10 *– The moveDown Myblock*

```
MyBlock(Name=moveDown, InParams=(v.NUM), OutParams=()) {

    Mu1:MathOp(Type=MUL, A=v.NUM, B=80)

    Ad1:MathOp(Type=ADD, A=Mu1.Res, B=900)

    Di1:MathOp(Type=DIV, A=Ad1.Res, B=49)

    Motor(Port=A, Dir=BK, Pwr=Di1.Res, PwrCtrl=ON,
      Dur=FOREVER)

moveDown}
```

To reproduce a different speed/space profile, it suffices to adapt the section of the controlling program that calculates the motor power. For the relation that implies a uniformly decreasing speed, you need to use the square root block. In the x_0 variable, the square root of the initial distance (fig. 5.4.11) is stored directly: therefore the first comparison allows the program to go on if the initial distance has a square root greater than or equal to 4. The other necessary modifications are rather straightforward and detailed in the NXT-GTD code (fig. 5.4.12).

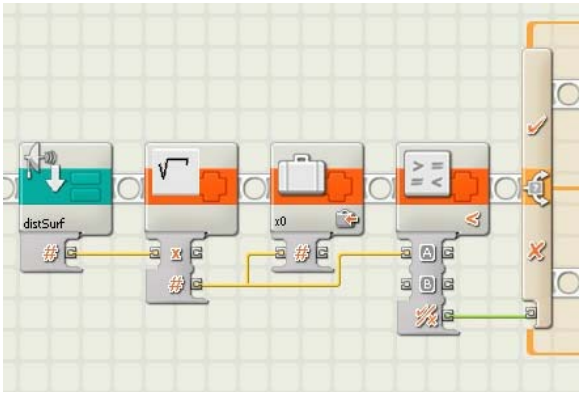

Fig. 5.4.11 – *The initial phase*

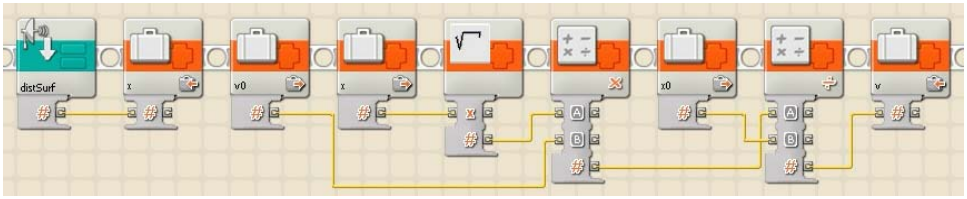

Fig. 5.4.12 – *The speed calculation*

```
Ds1:distSurf()

Sq1:SquareRoot(x1=Ds1.Result)

Var(Name=x0.NUM, Act=WR, Val=Sq1.Sqrt)

Cm1:CmpOp(Type=LT, A=Sq1.Sqrt, B=4)

Sw1: Switch(Ctrl=VAL, Type=LOGIC, Dis=ON, CondUp=TRUE,
          Val=Cm1.Res)

        Ds2: distSurf()
```

Var(Name=x.NUM, Act=WR, Val=Ds2.Result)

Vd1:Var(Name=v0.NUM, Act=RD)

Vd2:Var(Name=x.NUM, Act=RD)

Sq2:SquareRoot(x1=Vd2.Res)

Mu1:MathOp(Type=MUL, A=Sq2.Sqrt, B=Vd1.Val)

Vd3:Var(Name=x0.NUM, Act=RD)

Di1:MathOp(Type=DIV, A=Mu1.Res, B=Vd3.Val)

Var(Name=v.NUM, Act=WR, Val=Di1.Res)

This kind of experience can be repeated in another form with a simple bi-motorized vehicle (the classic *tribot*) and the sonar put in direction of the motion (fig. 5.4.13). The implementation of a speed/space profile will result as usual in a power-space profile that will produce a straight line motion, which follows the evolution deriving from the chosen profile, such as exponential for a linear profile and uniformly decelerated for a profile of a square root.

Fig. 5.4.13 – *The tribot*